

# Redes de Comunicação de Dados

## Protocolo de ligação lógica



Universidade do Porto

---

Faculdade de Engenharia

**FEUP**

**Trabalho e relatório elaborado por:**

Hugo Daniel Ferreira Almeida

João Paulo Castro Mendes

Nelson Jorge Silva Rodrigues

Ricardo Jorge Marques Veloso

ei00021@fe.up.pt

ei00022@fe.up.pt

nelson@fe.up.pt

ricardo.veloso@fe.up.pt

---

Faculdade de Engenharia da Universidade do Porto  
Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal

Abril 2004

## Introdução

Este trabalho tem como objectivo, a implementação de um protocolo de ligação lógica, e o seu teste, recorrendo a uma aplicação de transferência de ficheiros. A comunicação entre as duas máquinas será assíncrona e através da porta série RS232.

O protocolo desenvolvido apresenta um conjunto de interfaces de serviços à camada superior (llopen, llwrite, llclose). Neste caso, tais serviços serão utilizados pela aplicação de transferência de ficheiros.

Esta camada tem como funções principais, o estabelecimento/terminação da ligação lógica, o sincronismo das tramas, confirmação de recepção das tramas, controle de erros variados (rejeição de tramas, pedido de retransmissão, temporizações) e controlo de fluxo.

## Descrição

Antes da descrição promenorizada da implementação do emissor, convém dar um pouco de ênfase ao nível de mensagens trocadas no protocolo. Seguidamente será descrito o formato das tramas de controlo e informação utilizadas.

As tramas de controlo são constituídas por 5 bytes, flag inicial, campo de endereço, campo de controlo, block check character (BCC) e por uma flag final. As tramas de informação são compostas igualmente pelo cabeçalho similar, flag inicial, campo de endereço, campo de controlo, block check character, seguido de um número arbitrário de bytes correspondentes aos dados, por um BCC dos dados a enviar e pela respectiva flag indicadora de fim de trama.

Seguidamente serão apresentados os protótipos e funcionalidades das funções utilizadas pela aplicação, recorrendo ao uso de diagramas para uma melhor compreensão.

### Ficheiro emissor.c

Este ficheiro contém a função `main` relativa à aplicação emissor.

### Ficheiro receptor.c

Este ficheiro contém a função `main` relativa à aplicação receptor.

### Ficheiro ll.h

Neste ficheiro encontram-se os cabeçalhos das funções que estão implementadas e que a aplicação desenvolvida requer.

## Ficheiro ll.c

Além das funções gerais fornecidas, foram implementadas funções internas de suporte e ajuda de estruturação do código.

### **int llopen( int porta, int type )**

Esta função estabelece a ligação lógica. Recebe como argumentos a porta (0 | 1) e o tipo (RECEIVER | TRANSMITTER). Dependendo do tipo, chama uma das funções internas, `openConnection_trans` ou `openConnection_rcv`. Retorna o descritor da porta.

### **int openConnection\_trans( int porta )**

Abre a ligação lógica do lado do emissor. Recebe a porta e retorna o seu descritor.

### **int openConnection\_rcv( int porta )**

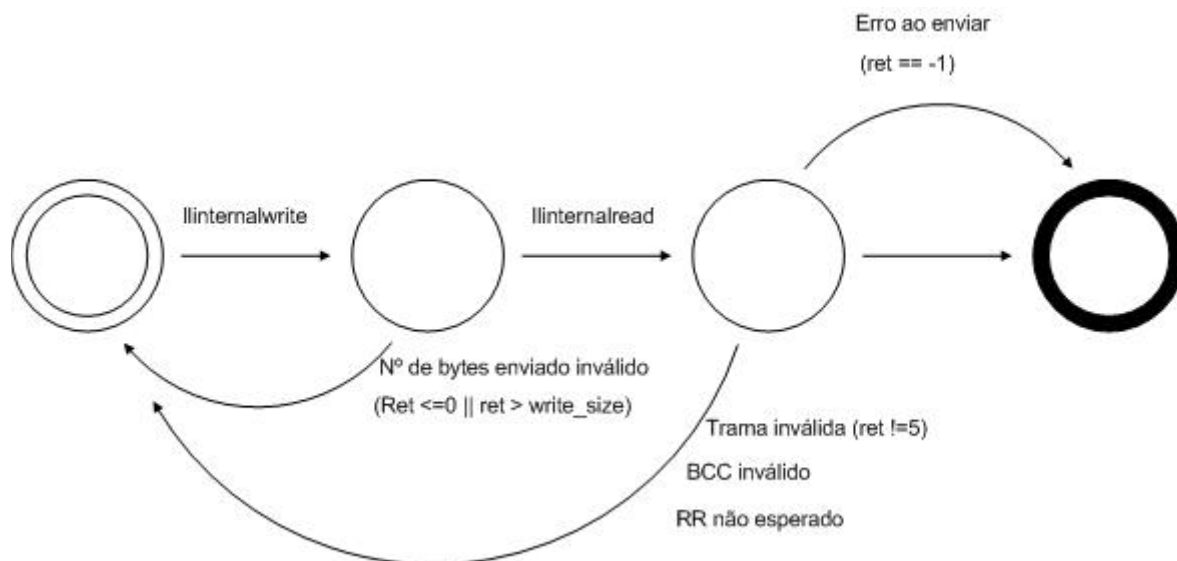
Abre a ligação lógica do lado do receptor. Recebe a porta e retorna o seu descritor.

### **int init( int porta, int vtime, int vmin )**

Esta função é chamada pelas duas funções anteriormente definidas. Recebe a porta assim como o número mínimo de caracteres a receber e o tempo de bloqueio. Vai ajustar estas as características à porta pretendida. Retorna o descritor da porta.

### **int llwrite( int fd, unsigned char\* buffer, int length )**

Função usada pela aplicação para enviar a informação desejada. Tem como argumentos o descritor da porta, previamente recebido em `llopen`, o buffer com os dados a escrever e o seu tamanho. Faz uso das funções `llinternalwrite` e `llinternalread`. Retorna o tamanho de bytes escrito ou -1 no caso de erro.



**int llread( int fd, unsigned char\* buffer, int length )**

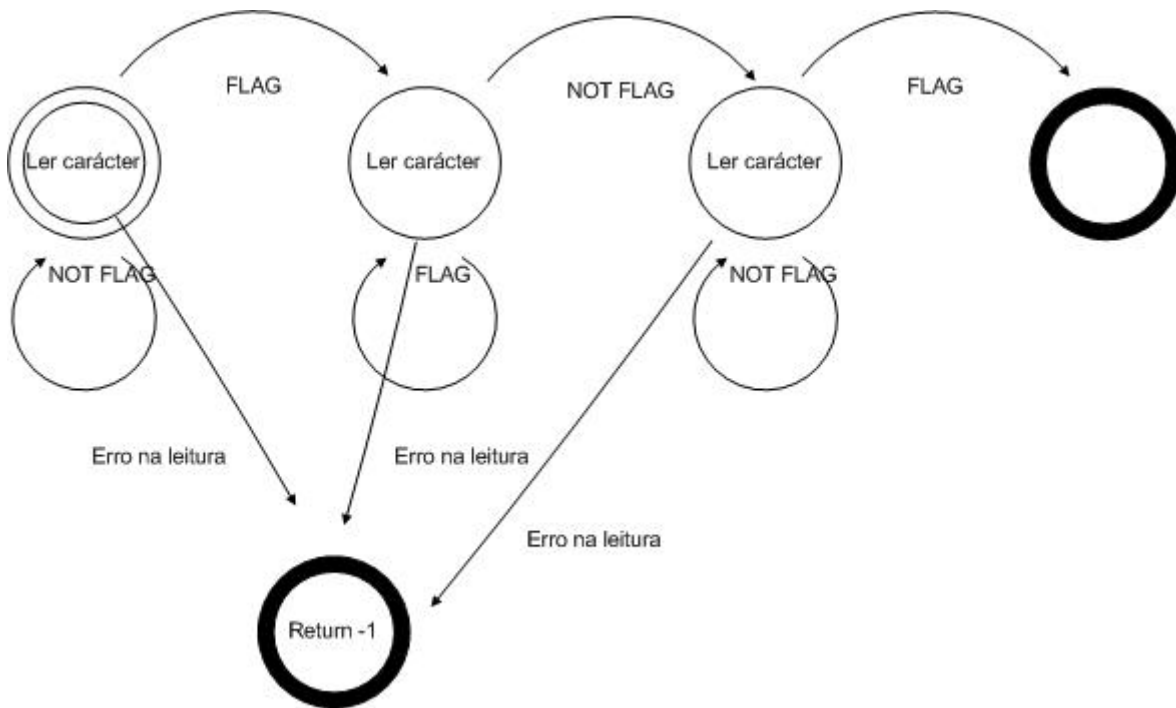
De forma análoga a llwrite, é usada para a leitura dos dados enviados. A trama recebida é armazenada em buffer. Faz uso das funções llinernalwrite e llinernalread. Retorna igualmente o tamanho de bytes lidos ou -1 se ocorrer um erro.

**int llinernalwrite( int fd, unsigned char \* buffer, int length )**

Função auxiliar utilizada por llwrite, acede a função bitStuffing e efectivamente escreve na porta. Recebe o descritor da porta, a trama a enviar e o seu tamanho. Retorna o número de bytes enviados.

**int llinernalread( int fd, unsigned char \* buffer, int length )**

Função auxiliary utilizada por llread. Lê uma trama do descritor fd, armazenando-a em buffer. Chama bitUnstuffing para reconstituição da trama desejada. Retorna o número de bytes lidos.



**void bitStuffing(const unsigned char \* buffer, int buffer\_len)**

Esta função tem como objectivo fazer o bit stuffing da trama a enviar. Os seus argumentos são o array que constitui a trama e o seu tamanho.

**void bitUnstuffing(const unsigned char \* buffer, int buffer\_len)**

Esta função tem como objectivo fazer o bit unstuffing da trama a receber. Recebe a o array que constitui a trama e o seu tamanho.

**int llclose( int fd )**

Fecha a ligação lógica estabelecida. fd indica o descritor da porta. Retorna o descritor da porta no caso de sucesso ou -1 em caso de erro.

**int closeConnection\_trans( int fd )**

Função interna, fecha a ligação do lado do emissor. fd é o descritor da porta recebido em llclose. Retorna os mesmos parametros da função anterior.

**int closeConnection\_rcv( int fd )**

Fecha a ligação lógica do lado do receptor. Tem igual funcionalidade que closeConnection\_trans.

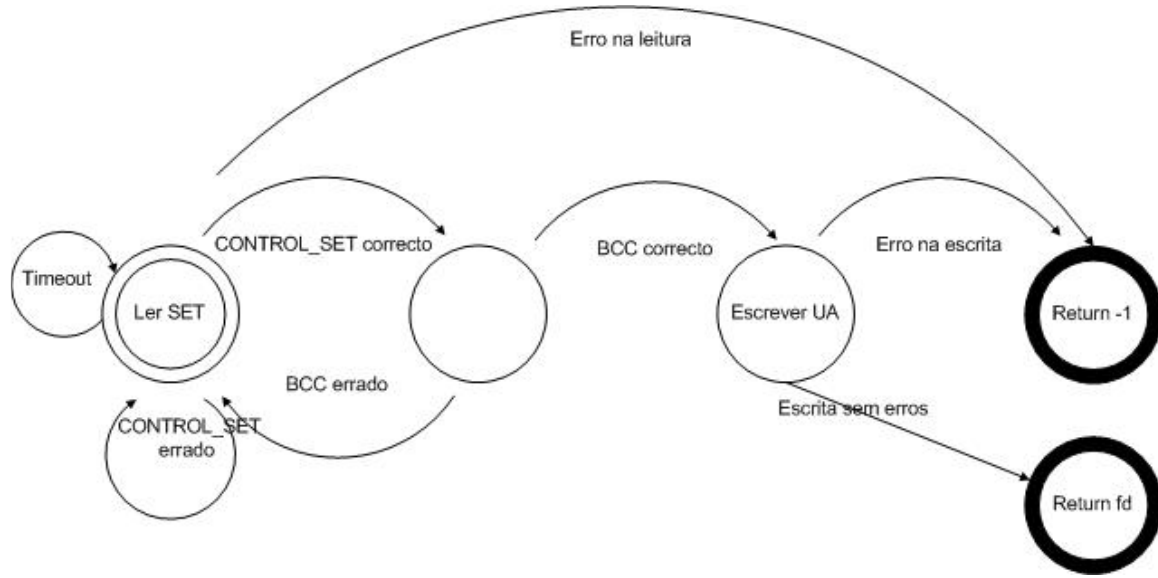
**int cleanup( int fd )**

Por último, esta função repõe as definições da porta identificada pelo descritor fd e fecha este.

## Máquina de estados

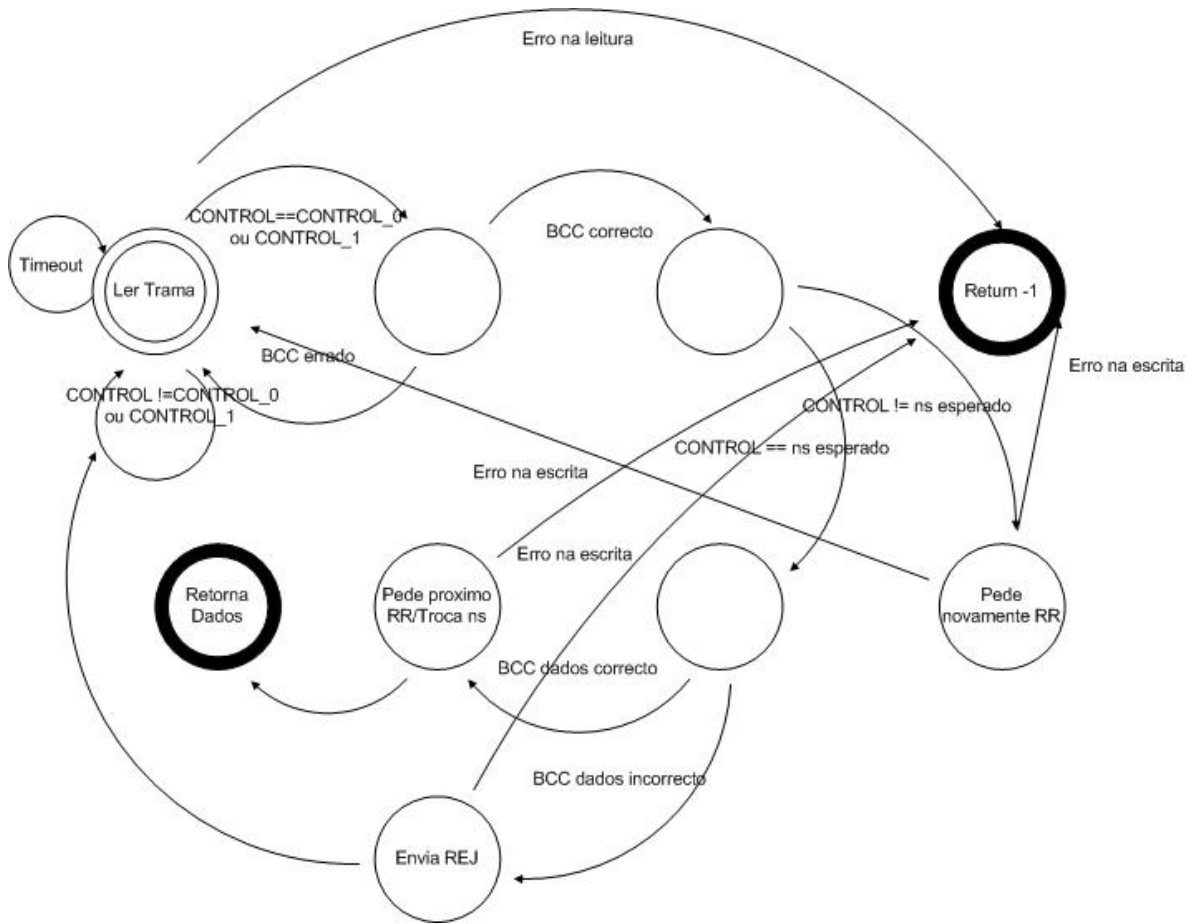
Para uma melhor compreensão, foi implementada uma máquina de estados do lado do receptor. Esta máquina de estados está dividida em três fases distintas:

### Fase de Estabelecimento da Ligação:



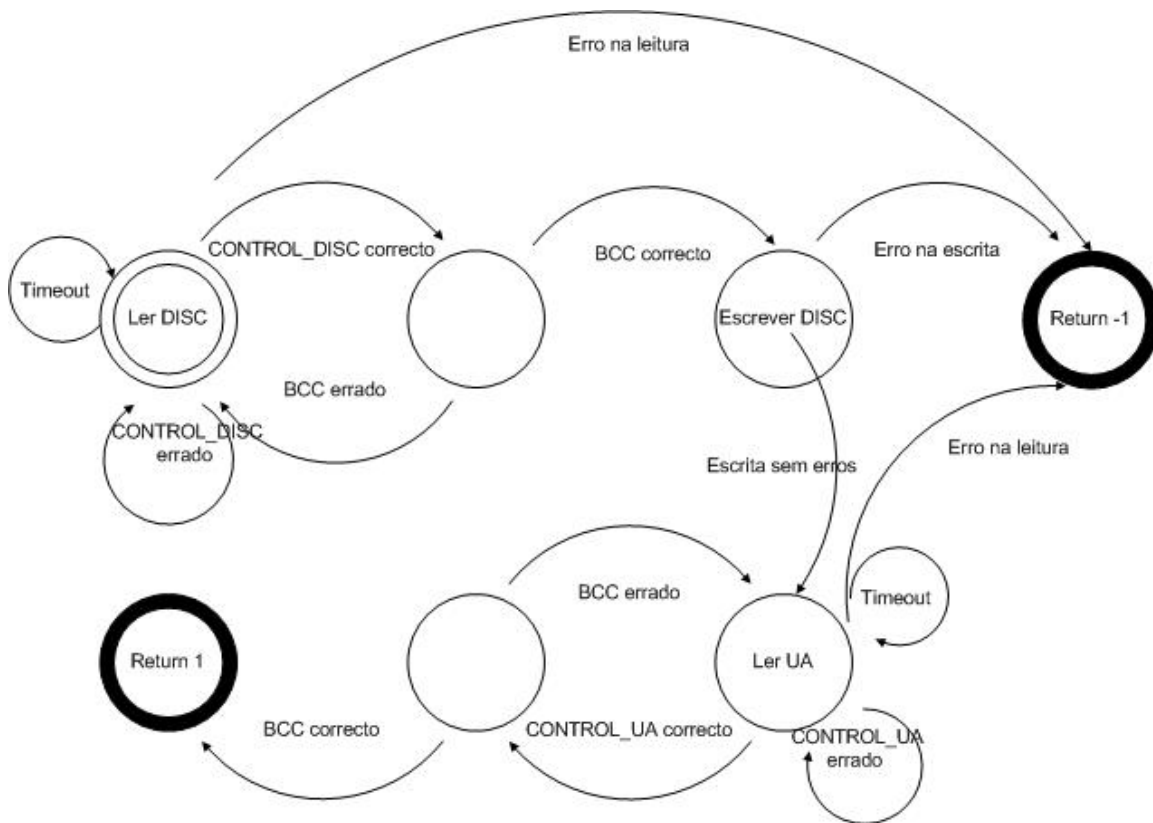
Nesta fase, o receptor fica à espera de uma trama, cujo campo CONTROL é o SET, indicando que está pronto para enviar os dados. Em seguida envia uma trama ao emissor, cujo campo CONTROL é UA, indicando que está pronto para receber os dados.

## Fase de Transferência de Dados:



Nesta fase, o receptor lê tramas de dados, verificando a sua consistência. A variável **ns** representada no diagrama pode tomar o valor de **CONTROL\_0** ou **CONTROL\_1**, sendo comutada conforme a necessidade.

### Fase de conclusão da ligação:



Nesta fase, o emissor, depois de chegar ao fim da transmissão de dados, envia uma trama ao receptor, cujo campo CONTROL é DISC, indicando ao receptor que acabou de enviar os dados e vai fechar a ligação. O receptor responde a essa trama com uma trama semelhante, ficando à espera de uma trama cujo campo CONTROL é UA, terminando assim a ligação.

## Modo de utilização

A aplicação de teste deverá receber dois parâmetros, porta (COM1 | COM2), e o nome do ficheiro a enviar, no caso do emissor, ou o nome do ficheiro a criar, no caso do receptor.

```
Ex: emissor COM1 pinguim.gif  
    receptor pingu.gif COM1
```

Para um debug completo do estado da camada lógica, na compilação deve ser usada a opção `-DDEBUG`.

```
Ex: gcc -DDEBUG -o emissor emissor.c ll.c  
    gcc -DDEBUG -o receptor receptor.c ll.c
```

## Considerações Importantes

Em seguida serão apresentadas algumas considerações tomadas no decorrer do trabalho:

- De modo ao receptor saber que o emissor acabou a transmissão do ficheiro, este último envia uma trama de dados vazia indicando end-of-file.
- Os timers usados no trabalho foram implementados através das opções da porta de série.
- O nome do ficheiro a enviar não é passado para o receptor. Ao executar a aplicação receptor, o utilizador deve indicar, como argumento, o nome que deseja que o ficheiro adquira na máquina de destino.

## Conclusões

O trabalho decorreu sem grandes problemas. O tempo de desenvolvimento foi o suficiente, não comprometendo de todo a conclusão deste.

Os objectivos propostos foram conseguidos.

## Referências

- Apontamentos das aulas teoricas